REMARKS

Claims 1-29 were presented for examination. All claims stand rejected. The Abstract has been amended. No claims have been amended, cancelled, or added. Thus, claims 1-29 are pending.

Objections to the Specification

The April 3, 2006 Office Action ("Office Action") objects to the Abstract for "its lack of disclosing the present invention." (Office Action, p. 2). While not conceding the objection, Applicants have amended the Abstract. As amended, the Abstract complies with MPEP § 608.01.

Claim Rejections under 35 U.S.C. §103

Mukherjee in view of Ando

Claims 1-2 and 4-29 are rejected under §103 (a) as being unpatentable over U.S. Patent Publication 2001/0034854 of Mukherjee ("*Mukherjee*") in view of U.S. Patent No. 6,519,730 of Ando et al., ("*Ando*"). For at least the reasons set forth below, Applicants submit that claims 1-2, and 4-29 are not rendered obvious by *Mukherjee* and *Ando*.

Claim 1 reads, in part, as follows:

> **comparing the result** from the selected instruction executed in the leading thread **to the result** from the selected instruction executed in the trailing thread; and
> (Emphasis added).

Thus, the emphasized portions of claim 1 recite comparing the result of executing a selected instruction in a leading thread with the result of executing the selected instruction in a trailing thread. Because *Mukherjee* and *Ando* fail to teach or suggest at least this limitation, claim 1 is patentable over the cited references.

Mukherjee, the primary reference, describes a simultaneous and redundantly threaded (SRT) processor and how the SRT processor checks uncached read instructions for transient faults. In describing transient fault tolerance for uncached reads, *Mukherjee* does not describe comparing the result of executing a read instruction in a leading thread with the result of a executing the read instruction in the trailing thread. Instead, *Mukherjee* describes the leading thread and the trailing thread separately writing a read request. (*Mukherjee*, para. 0041). Then a "compare circuit 148 preferably compares address from each related uncached read request from each thread." (*Mukherjee*, para. 0042). Thus, it is the read requests, including the read addresses, that are compared, <u>not the results of executing the uncached read request</u> as required by the above limitations of claim 1. The compared read addresses are not <u>results</u>. Instead, they are <u>inputs</u> or <u>operands</u> that are used with the read instruction.

Mukherhee never compares the results of executing the uncached read requests because only one of the uncached read requests is executed. "If these reads from each thread match exactly (the addresses are exactly the same), then only one of those uncached reads is allowed to proceed to read from main memory." (*Mukherhee*, para. 0042). The result of the read request is replicated for each thread. *Id.*

The Office Action states that comparison of the results of executing instructions in a leading and a trailing thread is disclosed at page 4, para. 0041, lines 8-10 of *Mukherhee*. (Office Action, p. 3). However, that portion of *Mukherhee* just describes comparing the uncached read requests, as described above. It does not describe comparing the results of read requests.

Therefore, *Mukherhee* does not teach or suggest claim 1's limitation of **"comparing the result** from the selected instruction executed in the leading thread **to the result from** the selected instruction executed in the trailing thread." (Emphasis added). *Ando* is not cited by the Office for purposes of disclosing the above limitation of claim 1. Therefore, claim 1 is not rendered obvious by the combination of *Mukherhee* and *Ando*.

Even if *Mukherhee* and *Ando* were read to teach or suggest all the limitations of claim 1, *Mukherhee* cannot be modified to practice the cited portions of *Ando* without making *Mukherhee* unsatisfactory for its intended purpose. MPEP § 2143.01V. One of the purposes of *Mukherhee* is to provide transient fault tolerance without sacrificing performance or increasing complexity. For example, *Mukherhee* teaches away from an SRT processor that executes the same program in different threads, while checking for transient faults upon the execution of each instruction. (*Mukherhee*, para. 0018). If the threads are lockstepped, *Mukherhee* objects to the overhead required by intensive verification of digital signals. *Id.* If one thread leads the other, *Mukherhee* still objects to the complications of verifying each thread's input and output. (*Mukherhee*, para. 0019).

Another purpose of *Mukherhee* is to simplify providing transient fault tolerance in processors that perform out-of-order execution of instructions. *Mukherhee* notes that, "The problem is further exacerbated if the one processor thread leads in overall processing location within the executed program. In this situation not only would the leading thread be ahead, but this thread may also execute the instructions encountered in a different sequence that the trailing thread." (*Mukherhee*, para. 0020).

To achieve transient fault tolerance while meeting the above objectives, *Mukherhee* only checks the correctness of memory stores and uncached reads. The cited *Mukherhee*

reference describes a technique for checking the correctness of uncached reads. Although *Mukherhee* does not describe fault recovery in detail – merely suggesting restarting each microprocessor thread at a point before the fault occurred. (*Mukherhee*, para. 0043). However, in the case of an uncached read, restarting at a point before the fault occurred could be as simple as reloading the read instruction and its read address operand – reloading a single instruction. If there was no further transient fault, the reads would match on the second load of the read instruction. There would be no need to re-execute or to maintain state regarding a series of instructions.

In contrast to *Mukherhee*, *Ando* describes a complex mechanism for transient fault recovery. *Ando* describes a processor with hardware that keeps track of state for instructions in a check point array. (*Ando*, col. 4, lines 5-9 and Fig. 1). A commit pointer points to the entry in the check point array where the state for the last correct, committed instruction is stored. (*Ando*, col. 4, lines 9-11). In the event of a transient fault, state corresponding to the last committed instruction is retrieved and an entire series of instructions is reexecuted. (*Ando*, col. 4, lines 60 to col. 5, line 10). Thus, *Ando* describes a transient fault recovery system with the type of overhead and complexity *Mukherhee* seeks to avoid.

Further, *Ando*'s complexity and overhead is unnecessary for transient fault recovery under *Mukherhee*. As discussed above, when practicing *Mukherhee*, recovery from a transient fault occurring during a uncached read may require only a reloading of the read instruction and its address operand. Thus, there is no motivation to modify *Mukherhee* to adopt the overhead and complexity of *Ando*. On the contrary, such a modification would

render *Mukherhee* unsatisfactory for its intended purpose of providing transient fault recovery while avoiding complexity and overhead.

In the event of a transient fault, the *Ando* fault recovery mechanism appears to just recover prior state and then re-execute a series of instructions in the original order. (*Ando*, col. 4, lines 60 to col. 5, line 10). *Ando*'s recovery mechanism is therefore contrary to *Mukherhee*'s goal of providing transient fault tolerance where a leading and a trailing thread practice out-of-order execution by executing the same series of instructions in a different order.

Therefore, for all the above reasons, modifying *Mukherhee* to use the recovery methods and hardware of *Ando* would make *Mukherhee* unsatisfactory for its intended purpose. The Office has therefore failed to make a *prima facie* case of obviousness. MPEP § 2143.01V.

Independent claim 11 similarly recites a limitation requiring a comparison of the results of executing an instruction in a leading thread with the results of executing the same instruction in a trailing thread. Therefore, claim 11 is patentable over the combination of *Mukherhee* and *Ando* for the same reasons that claim 1 is patentable over that combination of references.

Independent claim 14 recites as follows:

An apparatus comprising:

      leading thread execution circuitry to execute a leading thread of instructions;

      trailing thread execution circuitry to execute a trailing thread of instructions; and

      a history buffer coupled with the leading thread execution circuitry and the trailing thread execution circuitry to store information related to execution of a selected instruction from the leading thread of instructions, **wherein the**

information stored in the history buffer is used to restore an architectural state corresponding to a checkpoint if an execution fault is detected. (Emphasis added).

Thus, the emphasized portion of claim 14 recites a history buffer storing information used to restore an architectural state corresponding to a checkpoint. The information is used if an execution fault is detected.

The Office concedes that the above history buffer limitation is not taught or suggested by *Mukherhee*. (Office Action, p. 9). The Office contends that the history buffer limitation is disclosed by *Ando* at col. 3, lines 53-57. (Office Action, p. 9). However, that portion of *Ando* merely discloses restoring state for a "state storage device" and retrying execution of instructions from "the restored state." (*Ando*, col. 3, lines 55-57). There is no specific description of restoring an architectural state corresponding to a checkpoint. At best, the passage is vague concerning the type of state that is restored. Further, the cited passage does not specifically describe a checkpoint. Instead it appears that *Ando* checks for instructions for faults one-at-a-time and then commits an instruction if there is no error. When an error is detected, it simply retries the instructions from the last committed instruction. *Ando* does not appear to use checkpoints.

The Office also relies on col. 4 lines 5-18 of *Ando* as disclosing the above history buffer limitation. (Office Action, p. 10). This portion of *Ando* describes a "check point array" that stores a "previous correct state." (*Ando*, col. 4, lines 8-9). A "commit pointer" points to an entry in the check point array "at which the last correct instruction commit state is stored." (*Ando*, col. 4, lines 9-11). While this portion of *Ando* provides more detail regarding *Ando*'s state storage and retrieval mechanism, it does not specifically disclose a

history buffer used to restore an architectural state, such as, for example, a mapping of architectural registers to physical registers. *Ando* is vague about the contents of the check point array. This portion of *Ando* also does not specifically disclose restoring architectural state <u>corresponding to a checkpoint</u>.

However, even if the cited potions of *Ando* were read to disclose the above history buffer limitation, it is improper to modify *Mukherhee* to practice *Ando*'s state storage and retrieval techniques and hardware. As discussed above, modifying *Mukherhee* to practice what is described in *Ando* would render *Mukherhee* unsatisfactory for its intended purpose. MPEP § 2143.01V.

Therefore, the Office has failed to make a *prima facie* showing of obviousness. Claim 14 is patentable over the combination of *Mukherhee* and *Ando*. Independent claim 23 recites a system with the same history buffer limitation as claim 14. Claim 23 is therefore also patentable over the combination of *Mukherhee* and *Ando*.

Dependent claims 2, 4-10, 12, 13, 15-22, 24-29 all depend from one of the allowable independent claims 1, 11, 14, or 23. Therefore, they are also patentable over the cited references. MPEP § 2143.03.

Mukherjee in view of Ando and
in further view of Mukherjee Article

The Office rejected dependent claim 3 over the combination of *Mukherhee, Ando* and the publication "Detailed Design and Evaluation of Redundant Multithreading Alternatives," by Shubhendu S. Mukherjee, Michael Kontz and Steven K. Reinhardt in *Proc. 29th Int'l Symp. on Computer Architecture*, May 2002. However, the Office does not

14

cite the above article as teaching the limitations of claim 1. Claim 3 depends from patentable independent claim 1 and is therefore itself patentable. MPEP § 2143.03.

All pending claims are therefore patentable over the cited references.

CONCLUSION

Invitation for a Telephone Interview

The Examiner is requested to call the undersigned at (503) 439-8778 if there remains any issue with allowance of the case.

Request for an Extension of Time

The Applicant respectfully petitions for a one-month extension of time to respond to the outstanding Office Action pursuant to 37 C.F.R. § 1.136(a). A check for the fee under 37 C.F.R. § 1.17 is enclosed.
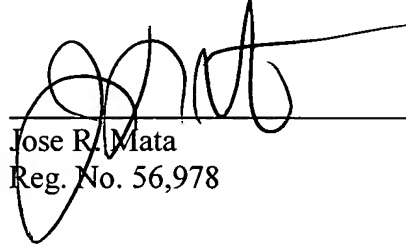
Charge our Deposit Account

Please charge any shortage to our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: _August 2, 2006_

Jose R. Mata
Reg. No. 56,978

12400 Wilshire Boulevard
7th Floor
Los Angeles, California 90025-1026
(503) 439-8778